Cricket Ball Prediction Model: A Conceptual Guide

Architecture Documentation

December 16, 2025

Abstract

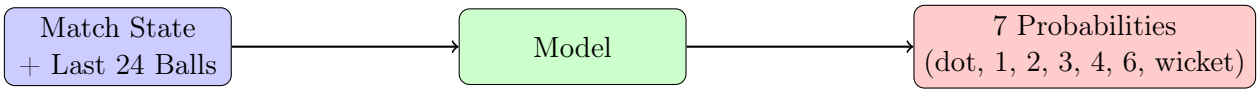
This document provides a conceptual understanding of the CricketPredictor model. Rather than focusing on equations, we emphasize **what goes in**, **what comes out**, and **what each component intuitively represents**. The goal is to understand the model well enough to rewrite it.

Contents

1 The Big Picture

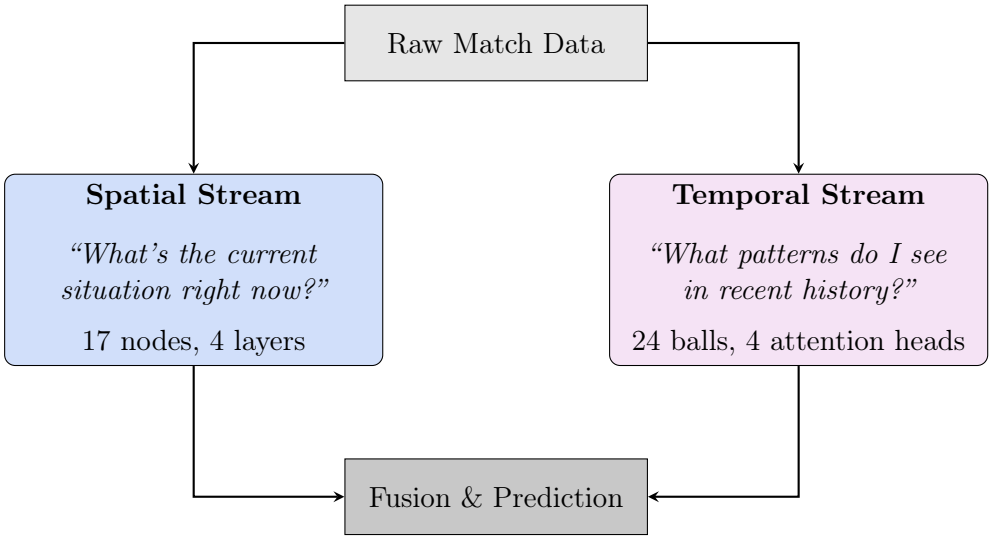
1.1 What the Model Does

Given the current state of a cricket match, predict what happens on the next ball:



1.2 Two Parallel Streams

The model processes information through two parallel pathways that answer different questions:



Output: [35% dot, 28% single, 10% two, 5% three, 12% four, 5% six, 5% wicket]

Figure 1: Dual-stream architecture: Spatial (current context) and Temporal (ball history)

2 The Spatial Stream: HierarchicalGAT

2.1 The Core Idea

Cricket has a natural hierarchy of factors that influence each ball:

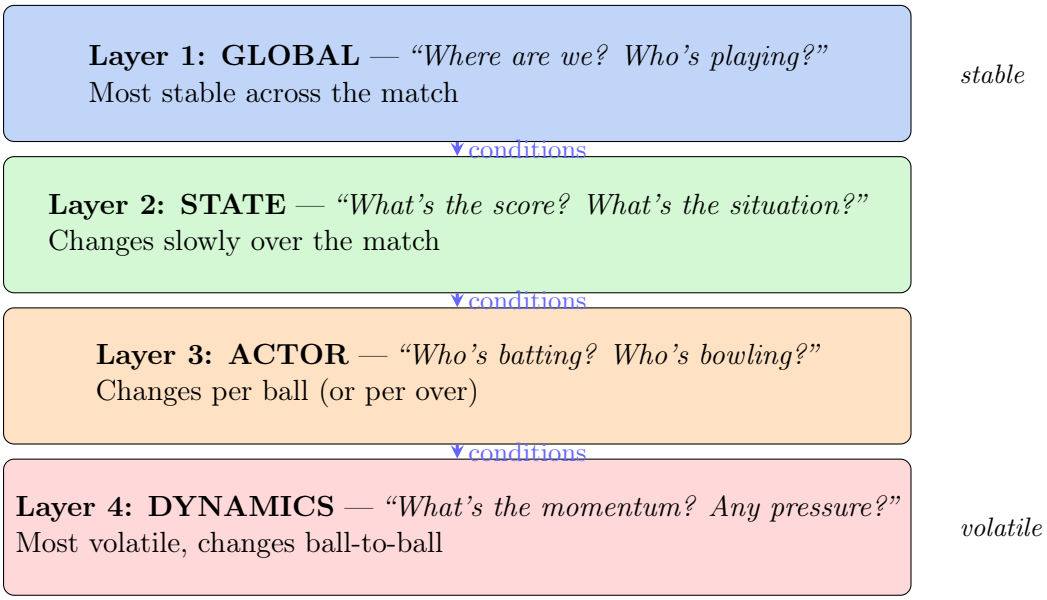


Figure 2: The hierarchical structure: higher layers condition lower layers

2.2 Key Insight: Top-Down Conditioning

Each layer receives context from the layer above it:

- **State** knows about the venue and teams (global context)
- **Actor** knows about the match situation (state context)
- **Dynamics** knows about the current batter/bowler (actor context)

This mirrors how a cricket expert thinks: "At the MCG (global), with India needing 50 off 30 (state), Rohit facing Bumrah (actor), after two dot balls (dynamics)..."

2.3 Layer 1: Global Context

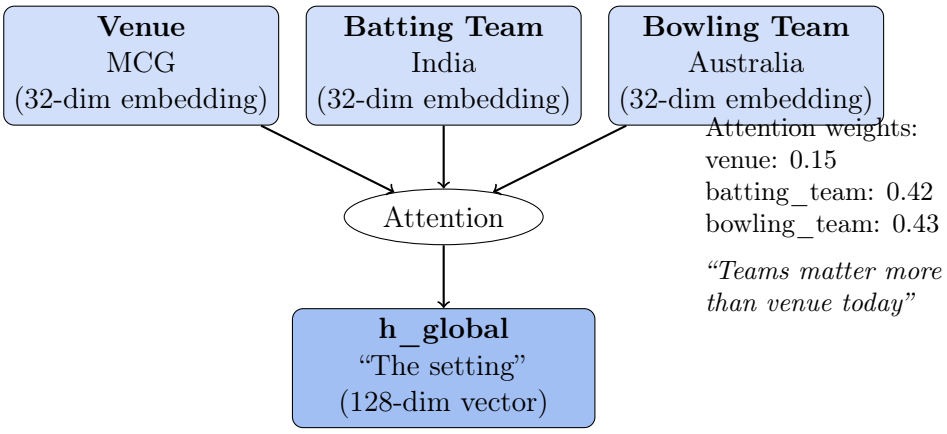


Figure 3: Layer 1: Global context aggregation

Inputs:

- `venue`: 32-dim learned embedding (encodes pitch type, ground size, typical scores)
- `batting_team`: 32-dim learned embedding (encodes team's batting style)
- `bowling_team`: 32-dim learned embedding (encodes team's bowling attack)

Output: `h_global` (128-dim) — represents "the setting" for this match.

Intuition: A learned query asks "What combination of venue and teams matters for this prediction?" and produces a weighted summary.

2.4 Layer 2: Match State

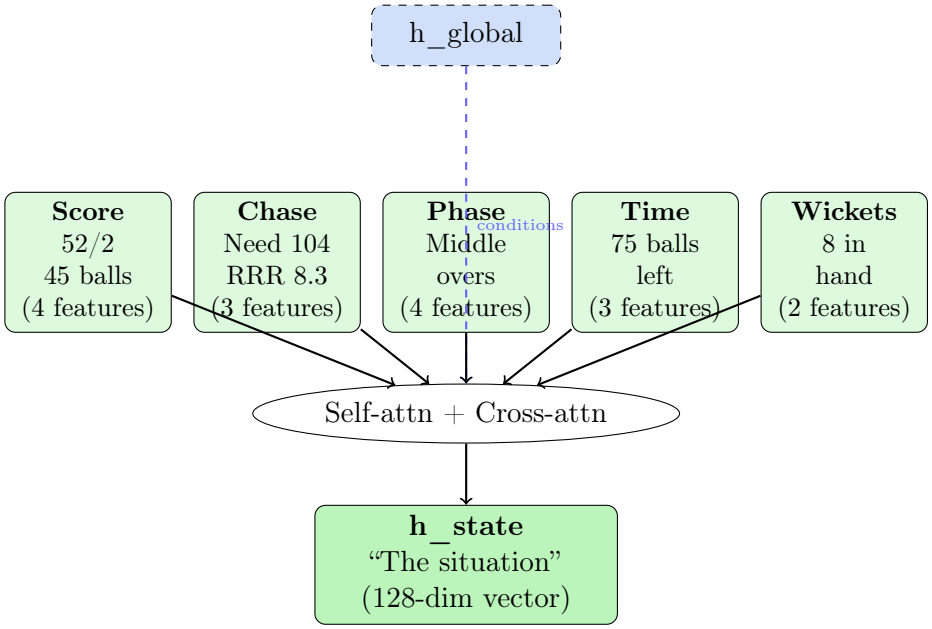


Figure 4: Layer 2: Match state with conditioning from global

Inputs (5 nodes, 16 features total):

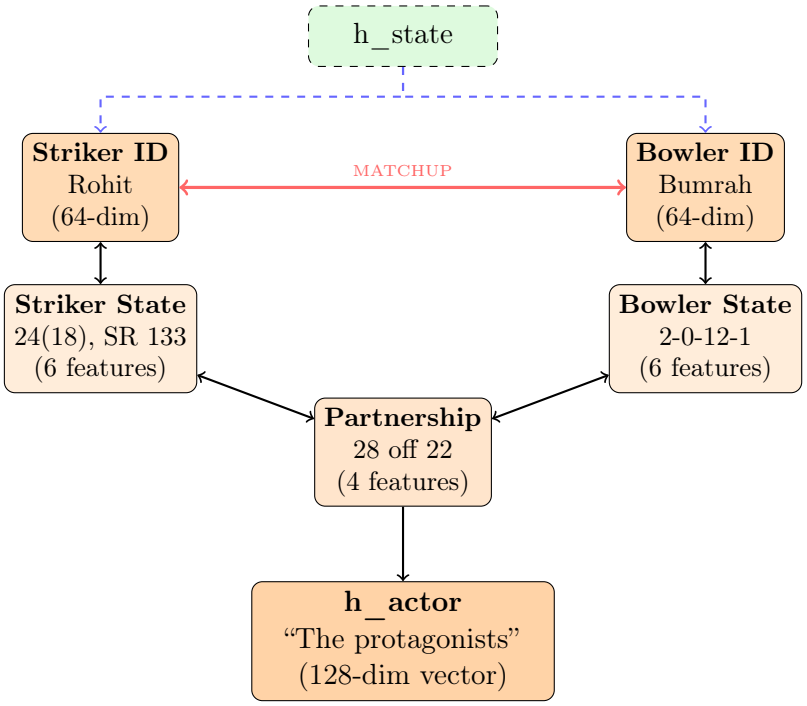
Node	Raw Features	Intuition
`score_state`	score/200, wickets/10, balls/120, innings	Current score status
`chase_state`	runs_needed/200, RRR/15, is_chase	Chase equation
`phase_state`	powerplay, middle, death, progress	Match phase
`time_pressure`	balls_left/120, urgency, is_death	Time remaining
`wicket_buffer`	wickets_in_hand/10, is_danger	Wicket cushion

Processing:

1. Self-attention: State nodes inform each other (e.g., chase relates to score)
2. Cross-attention: Nodes attend to h_global (situation interpreted through venue/teams)
3. Pool: Compress 5 nodes into one vector

Output: `h_state` (128-dim) — represents "the match situation."

2.5 Layer 3: Actor (The Matchup)



Graph Attention (GATv2): Nodes message-pass along edges.

Key edge: striker ↔ bowler

Figure 5: Layer 3: Actor graph with fixed edge structure

Inputs (5 nodes):

Node	What It Contains	Dimension
`striker_identity`	WHO is batting (learned embedding)	64
`striker_state`	HOW they're doing today (runs, balls, SR, dots, setness, boundaries)	6
`bowler_identity`	WHO is bowling (learned embedding)	64
`bowler_state`	HOW they're doing today (balls, runs, wickets, econ, dots, threat)	6
`partnership`	Current partnership stats (runs, balls, RR, stability)	4

The Graph Structure:

- Striker ID ↔ Striker State (who I am ↔ how I'm playing)
- Bowler ID ↔ Bowler State (same for bowler)
- **Striker ID ↔ Bowler ID** (THE MATCHUP — the key edge!)
- States ↔ Partnership (performance affects partnership)

Output: `h_actor` (128-dim) — represents "the current protagonists and their duel."

2.6 Layer 4: Dynamics

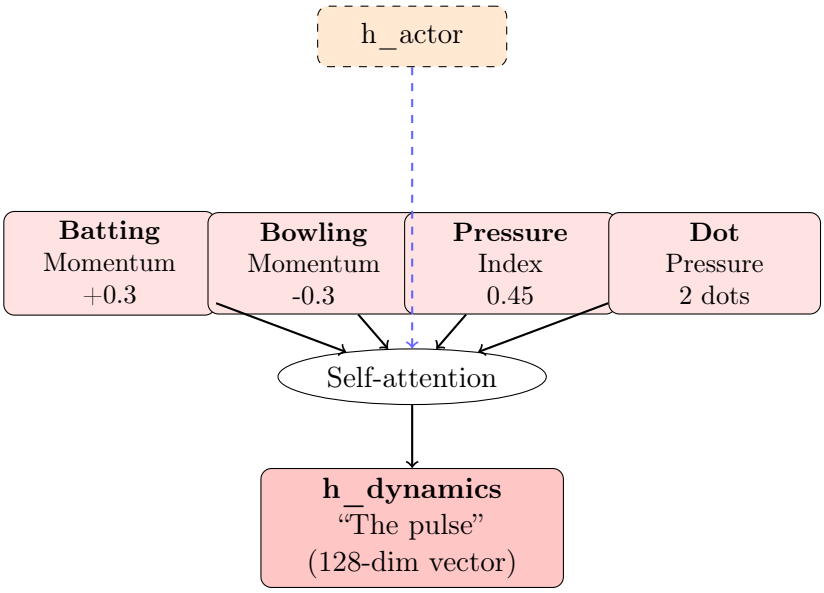


Figure 6: Layer 4: Dynamics — the most volatile layer

Inputs (4 nodes, 5 features total):

Node	Raw Value	Intuition
`batting_momentum`	(recent_RR / expected_RR) - 1	Are they scoring freely?
`bowling_momentum`	Negative of batting	Is the bowler on top?
`pressure_index`	Composite of wickets + RRR + stage	Overall pressure level
`dot_pressure`	consecutive_dots, balls_since_boundary	Building pressure

Output: `h_dynamics` (128-dim) — represents "the current momentum and pressure."

2.7 Combining the Four Layers

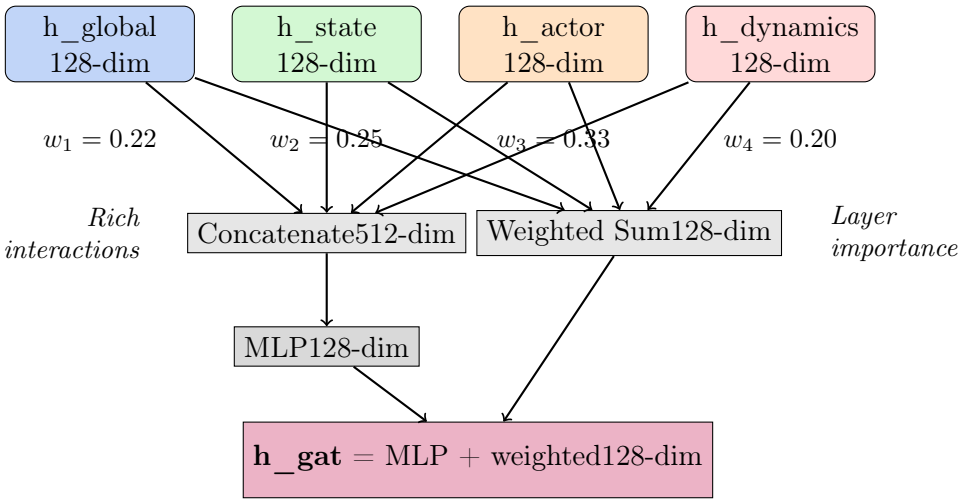


Figure 7: Fusion: both concatenation (for interactions) and weighted sum (for importance)

Why two fusion methods?

- **Weighted sum**: Learns which layer matters most (interpretable — "actor layer drives this prediction")
- **Concatenation + MLP**: Learns interactions between layers ("venue + actor combination matters")

Output: `h_gat` (128-dim) — the complete spatial representation.

3 The Temporal Stream

3.1 The Core Idea

While the spatial stream asks "What's the situation NOW?", the temporal stream asks "What patterns do I see in RECENT HISTORY?"

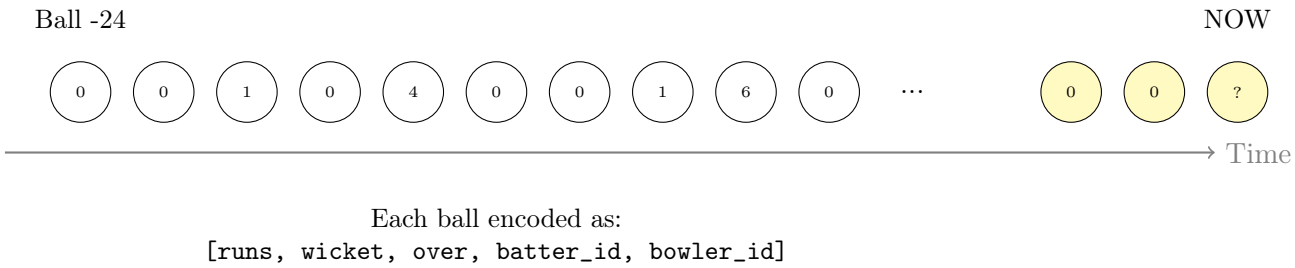


Figure 8: The temporal input: last 24 balls of history

3.2 Ball Embedding

Each historical ball is converted to a 128-dim vector:

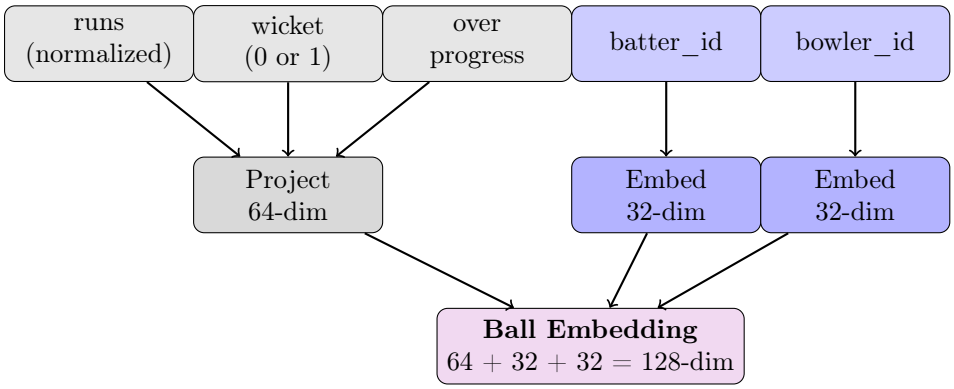


Figure 9: How each ball becomes a 128-dim embedding

3.3 Specialized Attention Heads

The key innovation: **4 attention heads with different biases**:

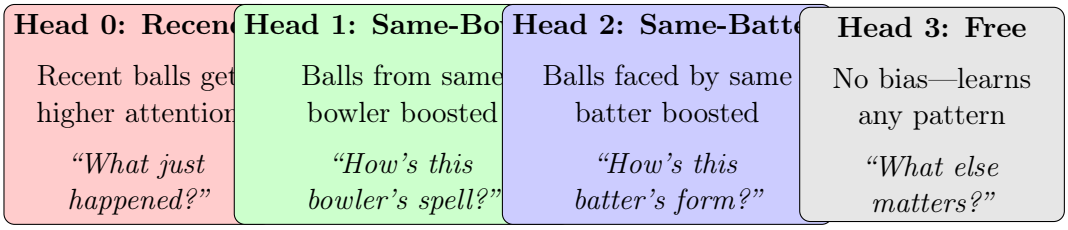


Figure 10: Four specialized attention heads encode cricket intuition

How the biases work:

- **Recency**: Adds $\alpha \cdot \frac{\text{position}}{24}$ to attention scores (ball 24 gets $+\alpha$, ball 1 gets 0)
- **Same-Bowler**: Adds β to attention when bowler matches (e.g., all Bumrah balls get $+\beta$)

- **Same-Batter**: Adds γ to attention when batter matches
- α, β, γ are **learned** during training

3.4 Query Token for Pooling

After the transformer layers, we need to compress 24 balls into one vector:

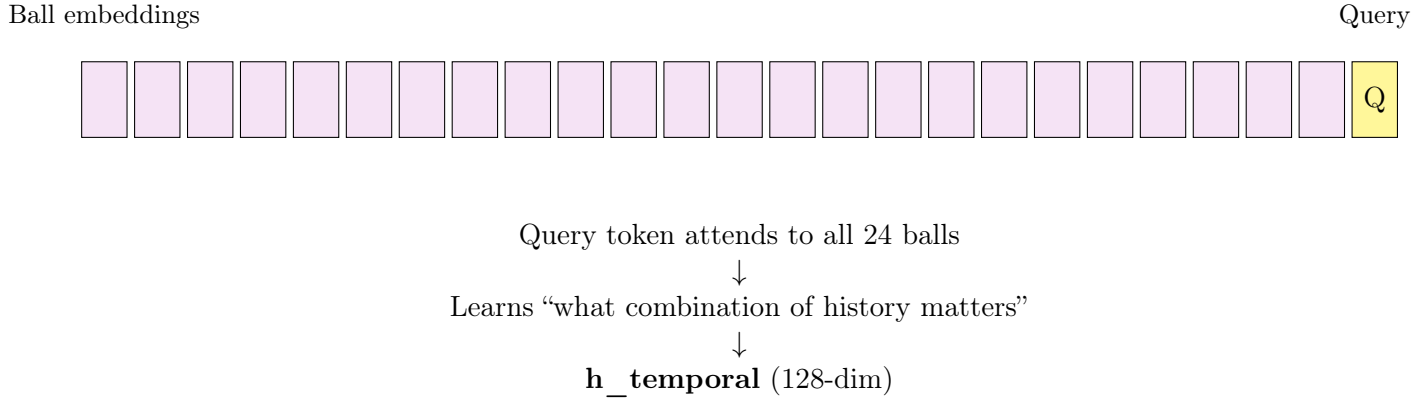


Figure 11: Learned query token pools the sequence

Output: `h_temporal` (128-dim) — the complete temporal representation.

4 Final Fusion and Prediction

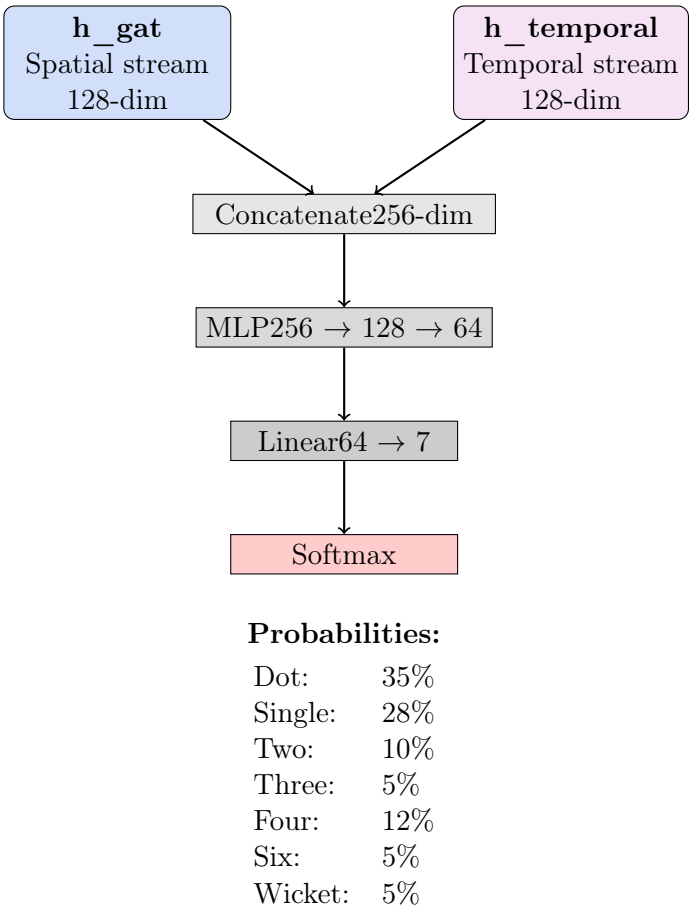


Figure 12: Final fusion: combine spatial and temporal, predict 7 outcomes

5 Complete Data Flow Summary

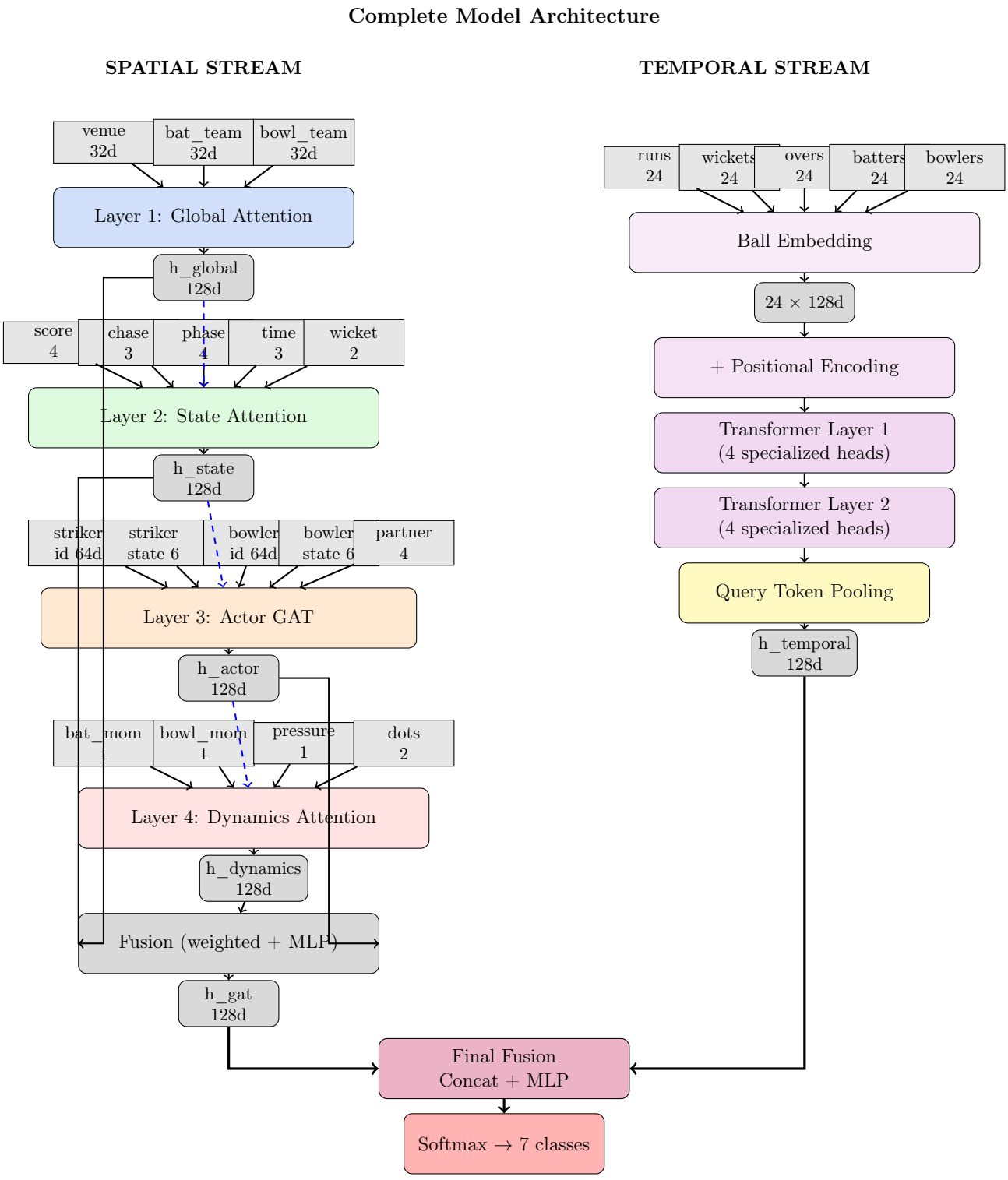


Figure 13: Complete architecture with all inputs, layers, and outputs

6 Feature Reference Tables

6.1 All 17 Spatial Nodes

Layer	Node Name	Input Dim	Features	Intuition
3*Global	venue	32	Learned embedding	Ground characteristics
	batting_team	32	Learned embedding	Team's batting DNA
	bowling_team	32	Learned embedding	Team's bowling style
5*State	score_state	4	score, wickets, balls, innings	Current score
	chase_state	3	runs_needed, RRR, is_chase	Chase equation
	phase_state	4	powerplay, middle, death, progress	Match phase
	time_pressure	3	balls_left, urgency, is_death	Time remaining
	wicket_buffer	2	wickets_in_hand, is_danger	Wicket cushion
5*Actor	striker_identity	64	Learned embedding	WHO is batting
	striker_state	6	runs, balls, SR, dots, set, bdry	HOW batter is doing
	bowler_identity	64	Learned embedding	WHO is bowling
	bowler_state	6	balls, runs, wkts, econ, dots, threat	HOW bowler is doing
	partnership	4	runs, balls, RR, stability	Partnership health
4*Dynamics	batting_momentum	1	recent_RR vs expected	Scoring momentum
	bowling_momentum	1	inverse of batting	Bowling control
	pressure_index	1	composite pressure	Overall pressure
	dot_pressure	2	consec_dots, since_boundary	Building pressure

Table 1: Complete feature reference for all 17 spatial nodes

6.2 Temporal Features (per ball)

Feature	Dimension	Description
runs	1	Runs scored on this ball (normalized)
wicket	1	1 if wicket fell, 0 otherwise
over_progress	1	Which over this ball was in (normalized)
batter_id	32	Embedding of batter who faced this ball
bowler_id	32	Embedding of bowler who bowled this ball
Total per ball	**128**	After projection and concatenation
Sequence length	**24**	Last 4 overs of history

Table 2: Temporal input features

7 Key Design Decisions

Decision	Current Choice	Alternative
Spatial vs Temporal	Parallel streams	Sequential (GAT per ball, then temporal attention)
Actor layer	GATv2 with fixed edges	Full attention or learned edges
Temporal heads	3 specialized + 1 free	All learned or all specialized
History length	24 balls (4 overs)	Full innings (more expensive)
Layer fusion	Weighted sum + concat	Only weighted or only concat
Graph structure	Treated as homogeneous	PyG heterogeneous graph

Table 3: Key architectural decisions and alternatives

8 What Each Output Represents

Output	Intuitive Meaning
h_global	"The setting" — venue + team context compressed into one vector
h_state	"The situation" — current match state (score, phase, pressure)
h_actor	"The protagonists" — who's batting, who's bowling, their duel
h_dynamics	"The pulse" — current momentum and pressure
h_gat	Complete spatial understanding of this moment
h_temporal	Patterns from recent history (last 4 overs)
Final output	7 probabilities: dot, 1, 2, 3, 4, 6, wicket

Table 4: Intuitive interpretation of each intermediate output